N91-20690

A Knowledge-Based approach to Configuration Layout, Justification and Documentation

F. G. Craig, D. E. Cutts, & T. R. Fennel
Boeing Computer Services
M/S JA-74
Huntsville Artificial Intelligence Center

C. M. Case & J. R. Palmer
Boeing Aerospace & Electronics,
Huntsville Division
M/S JY-33
P.O. Box 240002
Huntsville, AL 35824-6402

Abstract

This paper describes the design, development, and implementation of a prototype expert system which could aid designers and system engineers in the placement of racks aboard modules on Space Station Freedom. This type of problem is relevant to any program with multiple constraints and requirements demanding solutions which minimize usage of limited resources. This process is generally performed by a single, highly experienced engineer who integrates all the diverse mission requirements and limitations, and develops an overall technical solution which meets program and system requirements with minimal cost, weight, volume, power, etc. This "systems architect" performs an intellectual integration process in which the underlying design rationale is often not fully documented. This is a situation which lends itself to an expert system solution for enhanced consistency, thoroughness, documentation, and change assessment capabilities.

1.0 General Configuration Definition Issues

One of the major issues faced by any aerospace program is the need to consistently apply requirements, constraints, and resources to optimize the layout of equipment in an end item deliverable piece of hardware in the midst of changing environments. The change mandates can result from changing customer requirements, newly derived requirements, reduced program budgets, technological influences or personnel changes. All these changes tend to impact engineering processes, often rendering current approaches inappropriate or current solutions inadequate. In the remainder of this section we present a list (by no means exhaustive) of general issues which must be faced throughout a program's life cycle :

• Fleeting expertise : Turnover of domain experts represents a serious drain on program continuity and often causes work to be adversely impacted since significant portions of domain knowledge and program history often reside with individuals.

• Productive use of resources : Much layout work is both repetitive and resource intensive in nature. Allowing for automation of such repetitive tasks to be accomplished early in the process results in more resources being available for "real" engineering work to be performed later. This is usually a direct result of complimenting engineering expertise with tools which allow problems to be solved at a more abstract level and to off-load the repetitive portions of the task to the automated process. For example, engineering resources may be diverted to cost proposed changes, document accepted changes, and implement new procedures. Because of this type of required reaction, program continuity and productivity can be affected.

• Documentation of engineering rationale : All major programs have periodic requirements to review progress and to answer not only the question of "What has been done?", but also "Why was it done this way?", and "Why can it not be done this way?". The last two questions require the documentation, presentation and defense of engineering rationale. The problem is to provide a sound defense in areas where adequate documentation is generally missing, the expertise may have been lost, or rules may not have been codified, consistently applied, or documented.

• Multi-discipline inputs : Decisions made during the configuration process typically originate across several disciplines and organizational boundaries. Disciplines may or may not be aware of the impact of their decisions on other disciplines. These multi-disciplinary inputs to the engineering process highlight the need for a uniform approach to acquiring and representing those inputs.

• Explicit decision parameters and criteria : For engineering problems of any significant complexity, there is a need for the consistent application of clearly defined problem parameters and dynamic criteria to the engineering process. This is particularly true when these parameters and criteria come from various disciplines.

• Limited alternatives : Often the iterative engineering process is not fully utilized beyond a baseline "satisficing" solution (where the result is not optimal but merely satisfies most of the criteria). Little time is left to consider alternative analyses, configurations, or development paths. Better options may be overlooked because no tool/capability exists for quickly modelling and analyzing engineering alternatives.

• Problem of scale : Unfortunately, major program setbacks often occur because engineering solutions which worked well for small problems (or subsets of the larger problem) do not scale up well. This is particularly true when manual engineering approaches which were controllable and acceptable for the smaller problem are applied to large integrated programs.

As mentioned, the above list is not intended to be exhaustive, but is presented to serve as a reference for the next section.

2.0 Rack Layout Problem Description

As an initial test problem we have selected the Space Station Freedom module configuration task. This effort is similar to that required in many aerospace configuration layout applications in terms of complexity, constraints, and resources. It is above average in the number of expected major changes and long period of implementation. These factors make the configuration problem an ideal candidate for a knowledge based system.

The particular test domain area is that of rack placement aboard station modules. The racks provide the physical packaging for station services and functions. The objective of the rack placement process is to position a group of racks aboard modules in a configuration that minimizes utilization of resources, optimizes operational efficiency, and meets as many requirements and constraints as possible. The rack layout problem is representative of various configuration layout problems faced within many aerospace programs. Currently three other potential applications for this type of system have been identified within Work Package 1 of the SSFP, and it is expected that a number of additional spinoff applications will surface. Also, work performed on this project could be applied to areas external to the SSFP (other suggested areas include the outfitting of Commercial Aircraft and the Manned Mars Mission).

We are currently researching knowledge-based systems approaches to aid in this problem. The purpose of the research is to attempt to overcome the following perceived problems.

Fleeting Expertise : Currently, only one person in the Space Station program is identified as an "expert" on rack placement.

Claim : Development of an expert system to document the analysis, criteria, and engineering processes used by the expert will allow knowledge needed to solve the problem to be preserved and to be available for review by "non-experts".

Productive use of resources : The current manual approach to this process is quite time consuming and labor intensive. Rack layout reconfiguration for the station must be performed in step with other changes to the program. Due to lack of time, changes to rack configuration often represents an "acceptable" rather than an "optimal" solution.

Claim : The expert system is expected to significantly reduce the amount of time required to produce a new configuration. Additionally, the rules and procedures used by the system will be applied consistently through the automated program. Also, the expert system doesn't "forget" the rules or procedures during periods when the expert is busy with other task. Indeed, such an expert system could be used to train less skilled personnel to perform the task and can be used by the expert to explain the required analyses and procedures for the rack placement process.

Defense of engineering rationale : SSFP has a requirement for periodic reviews where design decisions must be justified.

Claim: In a rule-based system, the engineering rationale for a particular configuration is implicit in the set of rules used to generate that configuration. This engineering rationale provides placement justification and explanation. One of the objectives of the current work is to extract intelligible rationale from the set of rules used to generate the configuration.

Multi-discipline inputs: A large number of constraints exist between racks within and across modules. When these constraints are imposed on a large number of racks, a difficult constraint problem emerges. This problem is compounded by the fact that these constraints are imposed by different domain areas (such as power, thermal, cost, safety, etc.) and may be physical, functional, or operational in nature.

Claim : Experts from all applicable domain areas provide input to the rules and procedures used for the automated placement process. An additional advantage is that a unified approach to the acquisition, analysis, and representation of this domain knowledge can be developed and more easily verified by the experts from the various disciplines.

Explicit decision parameters & criteria : The lack of a uniform approach to explicitly identify applicable parameters and then consistently apply domain rules for rack placement hinders both the ability to quickly produce optimal rack layouts, and the ability to provide justification for a particular configuration.

Claim : The objects, rules, and associated parameters can be printed, queried interactively, and dynamically changed. This makes explicit the answers to questions such as:

What impact did the rule ....

"IF
the rack is rated as 'noisy' ,
THEN
don't place it near the crew sleeping quarters."

... have on the decision to place the rack?

Limited alternatives : Currently, analysis of rack configurations takes anywhere from several hours for the simplest changes to several weeks for more common changes. As expected, this does not leave much time for analyzing "What if...? situations."

Claim : Once the applicable parameters have been identified, and domain expertise has been captured, this expert system would support the ability to "tweak" priorities and constraints allowing engineers to analyze alternative configurations. The comparative "goodness" of rack configurations could be determined, and the tool could be used to suggest or support engineering change requests. Obviously, this does not imply that human expertise would no longer be needed. Rather it implies that more engineering analysis could be performed and human intuition could be used to fuller advantage by allowing the engineer to work at a higher level of abstraction.

Problem of scale : Currently the Phase 1 SSFP calls for only two modules and 4 nodes on the American portion of the program. Even this first phase of the program requires over 144 racks which must be assigned within a full range of physical, functional, and operational constraints. The multiplicity of constraints and the number of racks makes a manual approach to solving the problem nearly intractable.

Claim : While the number of racks and other "real world" objects is expected to remain relatively constant, it is anticipated that the number of constraint and control rules in the knowledge base will expand. No reliable data was available to estimate the bounds on the number of these rules. The tool selected for implementation set no upper bound on the size of the knowledge base (other than memory limitations). Speed was not a primary issue in this application, but reasonable response time was expected. The expert system incorporates domain expertise to control the focussing of rules which helps to limit the solution search (see the control layer in figure 1). Additional constraints can be easily added (or deleted) as knowledge about the racks and their interactions increases.

3.0 Implementation

The Nexpert expert system building tool from Neuron Data was selected for this project because it offered a number of desired features. Nexpert is a hybrid system supporting the representation of knowledge in objects and rules. It supports full inheritance and procedural attachment of methods as well as forward and backward rule chaining capabilities. It interfaces to user developed external routines as well as PC databases and spreadsheets. In addition, links to large databases such as Oracle and Informix are supported. Within this project we are currently a beta test site for a Hypercard/Nexpert "bridge" which allows communication between Nexpert and Hypercard facilities on the Macintosh II platform. Much of the explanation and training research is currently being performed using Hypercard. Nexpert is C based, runs on a wide range of hardware platforms, and offers a number of relatively inexpensive delivery options.

The prototype system software was implemented using a layered architecture to represent system knowledge (see Figure 1). This layered architecture separates different types of knowledge and aids in development, debugging and maintenance of the system. Chandrasekaran [Ref 3] proposes a similar architecture in which tools might be developed for "problem classes" such as diagnosis or design. He proposes that particular problems within these "problem classes" share similarities and that "generic" approaches to solving them might be appropriate.

Data resides in the lowest layer. The data is currently stored in a spreadsheet format, but facilities exist in the Nexpert tool to retrieve data from a number of sources including PC spreadsheets, PC databases, Oracle and Informix. Data is used to support the next layer representing objects and their associated attributes. These two bottom layers together might be thought of as Object-Attribute-Value (O-A-V) triplets. "Real world" entities such as modules, racks, standoffs, utilities, etc., are represented as objects in the system. Most of this type of knowledge was obtained directly from SSFP documentation. It should also be noted that this data might be obtained during the inference process or from some external source. This external source might well be an external routine which calculates a value and returns it to the object. This is analogous to attaching a "method" to as object.

The constraint layer contains all the constraint knowledge about the particular domain under consideration. This constraint knowledge is stored in rules and is a relatively "flat" knowledge base since this type of knowledge is concerned primarily with only a few "focal" objects (see Figure 2). This knowledge base consists of a collection of

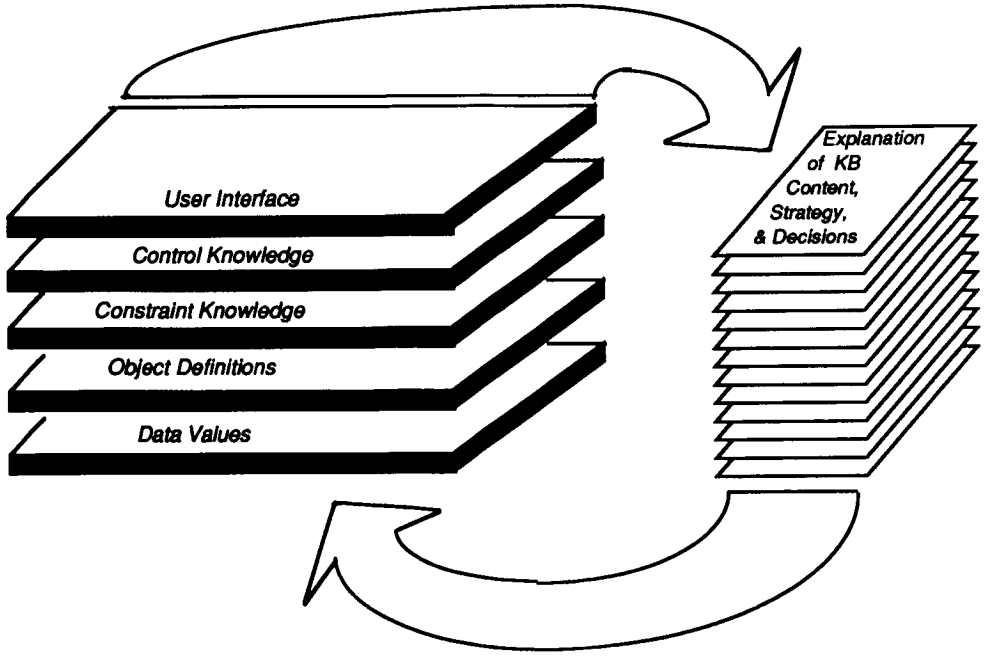

Figure 1 : The Expert System Architecture

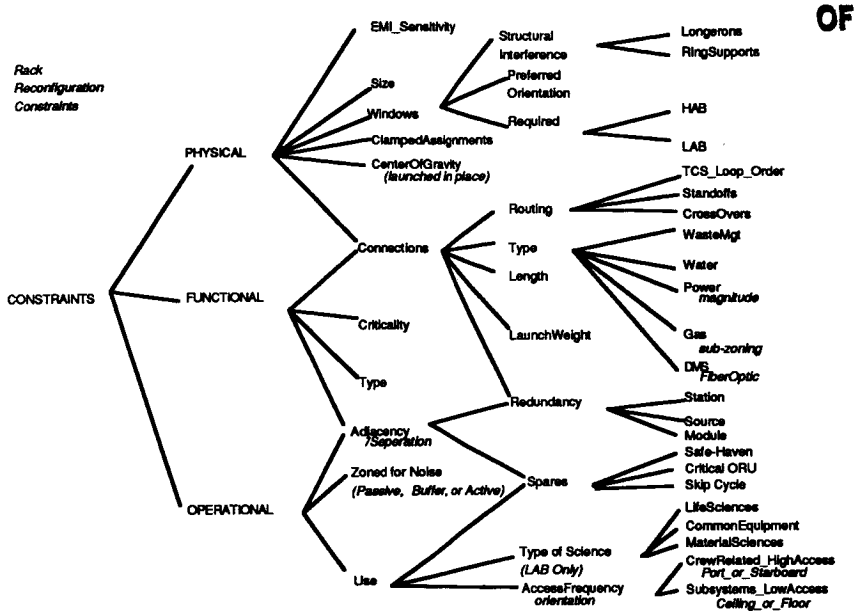

Figure 2 : Constraint rule hierarchy

"microscopic" rules to be used in solving the problem, and do not embody higher level "control knowledge" which a human expert would follow in applying the constraints.

The control knowledge (or meta knowledge) at the next level controls the direction of focus for the constraint knowledge. This level "prunes" the search space so that inappropriate constraint knowledge is not considered. Control knowledge is used to apply the constraint knowledge in much the same way a human expert would. An interesting offshoot of this project has been that the codification of this type of knowledge often helps to better define the problem solving process. This layer is also important in the explanation/justification of design decisions since explanations of design decisions made by the system need to be conveyed in much the same manner as a human expert's explanation.

The user interface represents the user's view of the system. For this application we have designed a "point and click" user interface in which the user manipulates racks within a module configuration. This interface provides input to the control layer about rack(s) to be moved. The control layer applies appropriate constraint knowledge at the next level.

The constraint layer, in turn, obtains needed information from the lower levels. The final result (**no constraints violated, "soft" constraints violated or "hard" constraints violated**) is passed to the user interface where the user can query the system about the particular decision and what support was used in making the decision.

The explanation/justification layer supports access to all the lower levels of the architecture. Queries can be made of objects, constraint knowledge or control knowledge. Explanations differ in content for these different layers and in level of detail based on the level of expertise of the user. A more detailed discussion of this layer can be found in [Ref 1].

4.0 Issues

As with any expert system project, there were a number of issues critical to success. Some of these issues are described below:

Expert availability: From the project's inception, a "domain expert" was identified and has been available at every step in the development process. This expert understands the problem to be solved and is able to articulate his method(s) for solving the problem.

Knowledge Acquisition: The data required for this project comes from both SSFP documents and domain experts. Traditional interview techniques with the primary rack placement expert as well as other experts in related fields have been very successful. Domain experts have recognized the potential utility of such an expert system and have supported it fully. In addition to interviews, the domain experts submitted "test cases" for the system to solve along with their conclusions/justifications as to why the move was good or bad (or could/could not be made). These test cases helped identify many weak areas in the system and helped to build the explanation facilities [Ref 1]. Early in the development process we began to use the system itself to acquire domain knowledge by running test cases against the system. This approach uncovered weak areas in the captured domain knowledge or incorrect assumptions. Thus, the tool itself has been used extensively in the acquisition process.

Verification and Validation: Test cases supplied by the domain experts as well as "working" interviews in which the system is used to test particular rack configurations have been used to test the system for correctness as well as its ability to provide meaningful decision justification. No work has been done to perform tests on the knowledge base for rule subsumption, rule contradiction, or cycles. Much of the system testing will continue to be empirical in nature.

5.0 Future work

One result of both the data acquisition and validation activities was that a larger number of people became aware of the project and began to look for ways to apply the work performed in the project to particular problems in their domain. As a result we anticipate that a number of spinoff projects will emanate from this IR&D work. Problems similar to the rack placement problem include resource allocation problems in which rack resource requirements are matched with resources supplied in the module to maximize the resource utilization. Another similar problem deals with the placement of payload racks (experiments, etc) within the lab module. This task must be performed repeatedly since experiments will continually be moved in and out. Another spinoff of this work may be in the training area. Much of the work being done to provide intelligent design justification and explanation could carry over into training new engineers on the SSF program.

As a component of a training system as well as design justification, we plan to interface this system with simulation systems to provide "deeper" justification or explanation by allowing the user to perform a simulation of a particular configuration during the design process. Adeli [Ref 2] reports on efforts to couple AI techniques with traditional mathematical techniques to aid in the engineering process.

6.0 Summary

Systems incorporating AI technologies to aid design will enhance the engineering process and will ensure that the decisions (and rationale behind them) are available in an intelligible format for future applications. Research in this area and prototype systems such as the one described here will help to clarify and define the engineering design knowledge capture requirements.

References:

[1] Fennel, T.R., et.al, "Graphical Explanation in an Expert System for Space Station Freedom Rack Integration", 5th Conference on Artificial Intelligence for Space Applications, Huntsville, AL, May 1990

[2] Adeli, H. & Balasubramanyam, K.V., "A Novel Approach to Expert Systems for Design of Large Structures", AI Magazine, Winter 1988, pp. 54-63

[3] Chandrasekaran, B. "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design", IEEE Expert, Fall 1986, pp. 23-30